UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/778,424	02/07/2001	Joseph C.H. Park	03226.037001; P5009	6879

22511 7590 04/21/2005

OSHA LIANG L.L.P.
1221 MCKINNEY STREET
SUITE 2800
HOUSTON, TX 77010

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 04/21/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/778,424	PARK, JOSEPH C.H.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *3/16/05*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-6,10,12-15 and 17* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-6,10,12-15 and 17* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *3/16/2005*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 3/16/2005.

As indicated in Applicant's response, claims 1, 10, and 17 have been amended. Claims 1, 3-6, 10, 12-15, and 17 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1, 3-4, 6, 10, 12-13, and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Intel® IA-64 "Architecture Software Developer's Manual", Jan. 2000 (IA-64), in view of Hwu et al., "A Framework for Balancing Control Flow and Predication", Proc. 30th Annual IEEE/ACM Intl. Symposium on Microarchitecture, Dec. 1997, pp. 92-103.; url: <http://citeseer.ist.psu.edu/august97framework.html> (hereinafter Hwu); further in view of Chen, USPN: 6,637,026 (hereinafter Chen).

As per claim 1, IA-64 discloses an extensible rule-based technique for optimizing predicated code comprising:

if-converting an abstract internal representation (e.g. §10.2.3.1, pg. 10-3 to pg. 10-4 – Note: analysis of instruction sequence by a compiler for branch prediction optimization implicitly discloses analysis of internal representation of program code);

testing a conditional code associated with a conditional condition (e.g. *cmp.ne p1, p0* - §10.3.1; *p1* ← *a != 0* - § 8.5; *conditions are false* - §10.3.1) and writing Boolean data to a

general register designated as destination register representing a predicate (see *target register* - §10.3.1; *p1 ← a != 0,(p1)add ...// if a!= 0 then add* - § 8.5 – Note: using a standard register to store predicate Boolean data is designated a generic register to make it a predicate register);

transforming the if-conversion to a machine representation (e.g. *assembler* - §10.2.3.1, pg. 10-3,4);

optimizing the machine representation based on a combination of a predetermined cover analysis (§10.2.1.2, pg. 10-2) and a predetermined replacement pattern such that a redundant instruction in the machine representation is eliminated (§10.2.3.1, pg. 10-3,4; §10.2.4, pg. 10-6->10-8 – Note: algorithmic case analysis for removing of extraneous cycles is equivalent to replacement pattern for eliminating redundant execution, e.g. branch instructions that would be otherwise not taken).

But IA-64 does not disclose that the above transformation includes eliminating predicates from the if-conversion representation, although IA-64 suggests not to use predicates under some conditions when applying the if-conversion/predication analysis (e.g. §10.2.4, pgs. 10-6-8). Hwu, in a method using hyperblock analysis to institute if-conversion analogous to the if-converting optimization by IA-64, also discloses the pitfalls of using predication when hardware resources, interference or repeated use of same resources and scheduling time frame are becoming a prohibitive factor (pg. 93, left col, line 5 to pg. 95, left col.), and ways to un-execute predication settings (associated with hyperblock analysis during the if-conversion) by applying the partial reverse if-conversion when the code to execute from the if-conversion is not desirable, or that hyperblock control flow lead to penalties (e.g. pg. 96, right col. to pg. 97, left col.). It would have been obvious for one of ordinary skill in the art at the time the invention was made

to apply the prevent execution of predicate instructions during if-conversion as analyzed and performed by Hwu and apply it to the proposed considerations by IA-64 for the reasons both mentioned by Hwu and IA-64 which are worsening the delay and resource usage during scheduling and execution of predicated code.

Hwu, however, does not explicitly disclose eliminating of predicates execution from the optimized conversion representation. In a method to optimize code using predication analogous to IA-64, Chen discloses potential resources, e.g. registers allocation, conflicts when setting up predicates in the code similar to the resources contention as mentioned by Hwu, and further discloses setting up virtual or redundant register mapping predicate based on which to validate or eliminate the execution of the intended predicate statement which can be oftentimes redundant (e.g. col. 1, lines 28-30; col. 2, line 35 to col. 3, line 56). It would have been obvious for one of ordinary skill in the art at the time the invention was made to apply the algorithmic approach to reverse the if-conversion as approached by Hwu (see above; pg. 99, Fig. 8) and additionally add the virtual statement code checking to establish as to whether a predicate execution is to be taken or eliminated as taught by Chen in order to enhance the teachings by IA-64 because of the same reasons causes as suggested by IA-64, or the interference effects by Hwu; and because this would expediently resolve register allocation conflicts due to predicate being competing for a same resource or redundantly based on same condition (see Chen: Background of invention).

As per claim 3, IA-64 does not expressly disclose eliminating a predicate defining instruction by interpretation; but Hwu, in a method using hyperblock analysis to institute if-conversion analogous to the if-converting optimization by IA-64, also discloses the pitfalls of using predication (re claim 1) and suggest ways to un-execute predication settings (associated

with hyperblock analysis during the if-conversion) by applying the partial reverse if-conversion when the code to execute from the if-conversion is not desirable, as in the case where hyperblock control flow lead to penalties (e.g. pg. 96, right col. to pg. 97, left col.); while Chen teaches eliminating of predicate execution when register allocation validating is checked as false (re claim 1). In view of IA-64 and the combined teaching of Hwu and Chen, the idea of optimizing at run-time is suggested, i.e. optimizing about exactly before runtime by many compilers (e.g. JIT) was strongly suggested by the architecture taught by IA-64. Further, official notice is taken that one method to alleviate runtime resource is to use interpretation of code when just-in-time optimization would have been useful any further was a known-concept in the art of runtime optimization, e.g. the well-known JIT compiler and Java optimizing virtual machine, at the time the invention was made. Hence, in case when the execution environment is having the interpretation capabilities of a optimizing JIT as mentioned above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the interpretation of code as a substitute for predicate in addition to the reverse if-conversion as suggested by Hwu or to the virtual predicate checking by Chen (in combination with IA-64) to avert unsafe use of resources due to predicate execution or compilation late optimization. One of ordinary skill in the art would be motivated to do this because, when the code to translate is a language which can be used by an just-in-time interpreter as mentioned above, enabling interpretation of code after if-converting as suggested by Hwu or after filtering of unjustified predicates as by Chen (in combination with IA-64) does not further complicate compilation resources when every possible optimization process would have been exhausted and that only size-simplified, branch-free and fault-free instructions are left to execute just as evidenced in the

teachings from the notice from above and in conjunction with the analysis and decision making in Hwu's algorithm for reverse if-converting hyperblocks (see Hwu: pg. 97-100).

As per claim 4, IA-64 does not specify eliminating a guard predicate of a safe instruction by speculation but discloses using both speculation and predication (e.g. § 2.6, pg. 2-4 to pg. 2-5; § 8.4-5, pg. 8-4 to pg. 8-6); and discloses how using predication can be inappropriate when overlapping memory resources are identified (re claim 1). Hwu, as in claim 3, further discloses applying an analysis to ensure that a predication is resolved and free of penalty (resources, cycle count, interaction, dangling latencies - pg. 97-100) by applying a reverse if-conversion to optimize resource usage at scheduling time; and Chen discloses predicate-checking provision to control register conflicts when resolving resource safe aspects of a predicate statement execution (re claim 1). Official notice is taken that using speculation when it is determined that a code is free of ambiguous control flow or data dependency as suggested by IA-64 was a known-concept at the time the invention was made. Hence, in view of the above notice and suggested teachings by IA-64, Hwu and Chen, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the speculation of code to substitute for some predicated execution as suggested by Hwu (in combination with IA-64) to avert unsafe predicate execution when it is determined that there is no ambiguous dependency of control or data flow as taught by Official notice from above and by Hwu's techniques. One of ordinary skill in the art would be motivated to do this because enabling speculation of code after the predicates resources dependency analysis as suggested by Hwu's correct scheduling (in combination with IA-64/Official notice) further alleviates execution resources from correctly predicting of data/control flow just as intended by the optimization proposed by IA-64 (combined with Hwu's teachings),

notably when predicate implementation tends to increase size of code (re IA-64 – pg. 10-6 to pg. 10-8).

As per claim 6, IA-64 does not disclose using reverse if-conversion but this limitation would have been obvious in view of IA-64's teachings about the pitfalls in predication (re claim 1) and the rationale using Hwu's partial reverse if-conversion method and using Chen's elimination of redundant, register-unallocated or unjustified predicates based on some checking.

As per claim 10, this apparatus claim including means performing the step limitations that correspond to those of claim 1 and is rejected with the corresponding rejection as set forth therein.

As per claims 12, 13, and 15, these claims correspond to claims 3, 4, 6 respectively, and are rejected with the rejections as set forth therein.

4. Claims 5, 14, and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Intel® IA-64 "Architecture Software Developer's Manual", Jan. 2000 (IA-64), in view of Hwu et al., "A Framework for Balancing Control Flow and Predication", 1997, and Chen, USPN: 6,637,026; as applied to claim 1 (for claim 5), 10 (for claim 14) above, and further in view of Schlansker et al., USPN: 5,999,738(hereinafter Schlansker).

As per claim 5, IA-64 (combined with Hwu/Chen) does not disclose eliminating a guarding predicate of an unsafe instruction by compensation. IA-64 and Hwu, as in claim 3, discloses using analysis to determine resources dependency, cycle counts and interdependent scheduling (re claim 3-4) for averting using of penalty-prone or inefficient predicate-guarded block of instructions; while Chen discloses eliminating of a predicates execution based upon resolving resource allocation by means of redundant/virtual validation predicates (re claim 1).

Schlansker, in a method to selectively execute fully or partially resolved predications analogous to the partial reverse if-conversion by Hwu or register mapping by Chen, discloses compensation code to stand for predicate code when the latter undergoes resolution as to whether it can fall-through or not (Fig. 10); hence has suggested substituting predicate instruction in case such predicate result in non fall-through resolution. Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the use of compensation code to substitute for some predicate guard resolution leading to unsafe code as suggested by Hwu/Chen (in combination with IA-64) to avert unsafe predicate execution when it is determined that there potential unresolved predicated control flow can result in a penalty as taught by Schlansker. One of ordinary skill in the art would be motivated to do this because enabling compensation code to cover for predicate ill-selected paths as suggested by Schlansker's provision further alleviates execution resources from having to remedy for mispredicted path or paths resulting in wrong direction as originally intended by the optimization proposed by IA-64 (combined with Chen/Hwu's teachings), notably when predicate implementation tends to increase size of code (re IA-64 – pg. 10-6 to pg. 10-8).

As per claim 14, this claim corresponds to claim 5, and is rejected with the rejection as set forth therein.

As per claim 17, IA-64 discloses an extensible rule-based technique for optimizing predicated code comprising:

if-converting an abstract internal representation (e.g. §10.2.3.1, pg. 10-3 to pg. 10-4);

testing a conditional code associated with a conditional condition (e.g. *cmp.ne p1, p0* - §10.3.1; *p1* ← *a != 0* - § 8.5; *conditions are false* - §10.3.1) and writing Boolean data to a

general register designated as destination register representing a predicate (see *target register* - §10.3.1; *p1 ← a != 0,(p1)add ...// if a!= 0 then add* - § 8.5);

transforming the if-conversion to a machine representation (e.g. *assembler* - §10.2.3.1, pg. 10-3,4);

eliminating the predicates from mapped if-conversion (e.g. §10.2.4, pgs.10-6-8); and

optimizing the machine representation based on a combination of a predetermined cover analysis (§10.2.1.2, pg. 10-2) and a predetermined replacement pattern such that a redundant instruction in the machine representation is eliminated (§10.2.3.1, pg. 10-3,4; §10.2.4, pg. 10-6—10-8 – Note: algorithmic case analysis for removing of extraneous cycles is equivalent to eplacement pattern for eliminating redundant execution, e.g. branch instructions that would be otherwise not taken).

But IA-64 does not disclose that the transformation includes eliminating predicates from the if-conversion representation. But this limitation has been addressed in claim 1.

Nor does IA-64 expressly disclose eliminating a predicate by interpretation as recited in claim 3, by speculation as recited in claim 4, by compensation as recited in claim 5, by reverse if-conversion as recited in claim 6; but these limitations have been addressed with the corresponding rejections as set forth therein, respectively.

Response to Arguments

5. Applicant's arguments filed 3/16/2005 have been fully considered but they are not persuasive. Following are the Examiner's observations in regard thereto.

Rejections under 35 USC § 103:

(A) Applicant has submitted that IA-64 is silent as to condition codes and general register writing (Appl. Rmrks, pg. 9, 3rd para). The claim does not provide specific teaching as to what particularly a 'condition code' consists of. A code having a conditional connotation so that the reading of which yields some consequence or action based on such reading is interpreted as condition code. The example provided by IA-64 by the instruction of the likes of *cmp ne* or *cmp.eq* then *(p1) br.cond* (see §10.2.2) entails necessarily a code being read during a comparison founded on a given Boolean state representing a condition to match, e.g. equal to or not equal to; and such conditional state reads on condition code. The rejection also shows as a consequence of such *cmp* instruction and reading leading to getting a condition in a form of a Boolean status; such status being written back to register such as *p1*. Regarding the register writing limitation, it is noted that IA-64 provides a register, say *p1*, designed to store Boolean result as a consequence to a read and comparing of status bit in registers, such status set as a condition for effecting a subsequent instruction; and that reads on the claimed feature. Hence, the Applicant's argument is not persuasive for reading into the claim more than what the claim really amounts to in terms of reasonable teachings as being broadly interpreted.

(B) Applicant has submitted that Hwu is silent as to condition codes or general register writing (Appl. Rmrks, pg. 9, 4th para). The argument is not persuasive because Hwu is used to address another limitation which the 103 rejection is purported to address. By pointing out deficiencies of references taken separately from the context of a prima facie type of obviousness rejection, the Applicant fails to establish appropriate grounds for defeating the purpose of the combination as set forth in this type of rejection. It is required that the argument have to point out specifics as to why the prior art teachings such as combined would yield undesirable results

or adverse effects. Besides, the base reference IA-64 already provides teaching to address what Applicant calls 'condition code' and to address what is required as writing to a register based on the testing (see rejection); and until the claim makes it very specific as to exactly what these limitations are all about in undisputable specificity; the teaching by IA-64 has fulfilled the features at issue, features for which Hwu is not brought in to fulfill.

(C) Applicant has submitted that Chen is silent as to condition codes or general register writing(Appl. Rmrks, pg. 10, top). Chen and Hwu are brought in to address some teaching as to why it is beneficial for effecting elimination of predicate instructions which IA-64 does not appear to disclose. The arguments are aiming at each reference taken individually hence have not succeeded in pointing out how the combined teachings as set forth in the rejection would fail to distinguish from the claimed invention. The rejection has shown what is missing in IA-64 and what Chen and Hwu's suggested approach would enable IA-64 to eliminate redundant execution; and alleviate resources.

For the above reasons, the rejection will stand as set forth.

Conclusion

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before

using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
April 17, 2005

KAKALI CHAKI
SUPERVISORY PATENT E
TECHNOLOGY CENTE